SISTEMAS DISTRIBUIDOS – GRUPO A

ENTREGABLE2 LAB

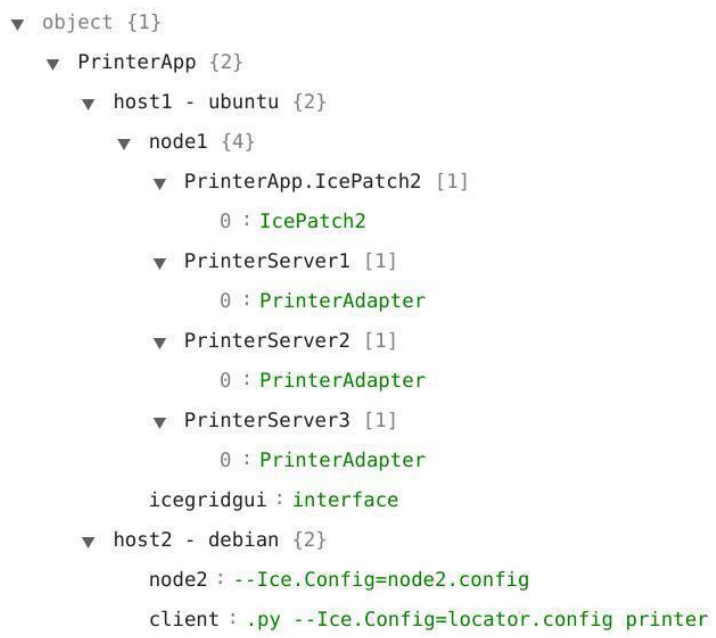
INDICE

REPOSITORIO GITHUB
https://github.com/jupcan/zeroc-ice

COMPONENTES DEL GRUPO
Juan Perea Campos

1. DOCUMENTACIÓN ENTREGABLE2

1.1 DIAGRAMA DE LA ARQUITECTURA DE LA SOLUCIÓN



Hay dos hosts diferenciados, uno de ellos mi ordenador portátil con ubuntu y el otro una máquina virtual debian desplegada con la imagen que los profesores nos facilitaron para poder simular el funcionamiento de la aplicación distribuida como si se tratara de otro ordenador independiente a pesar de estar ambos ejecutados en la misma máquina.

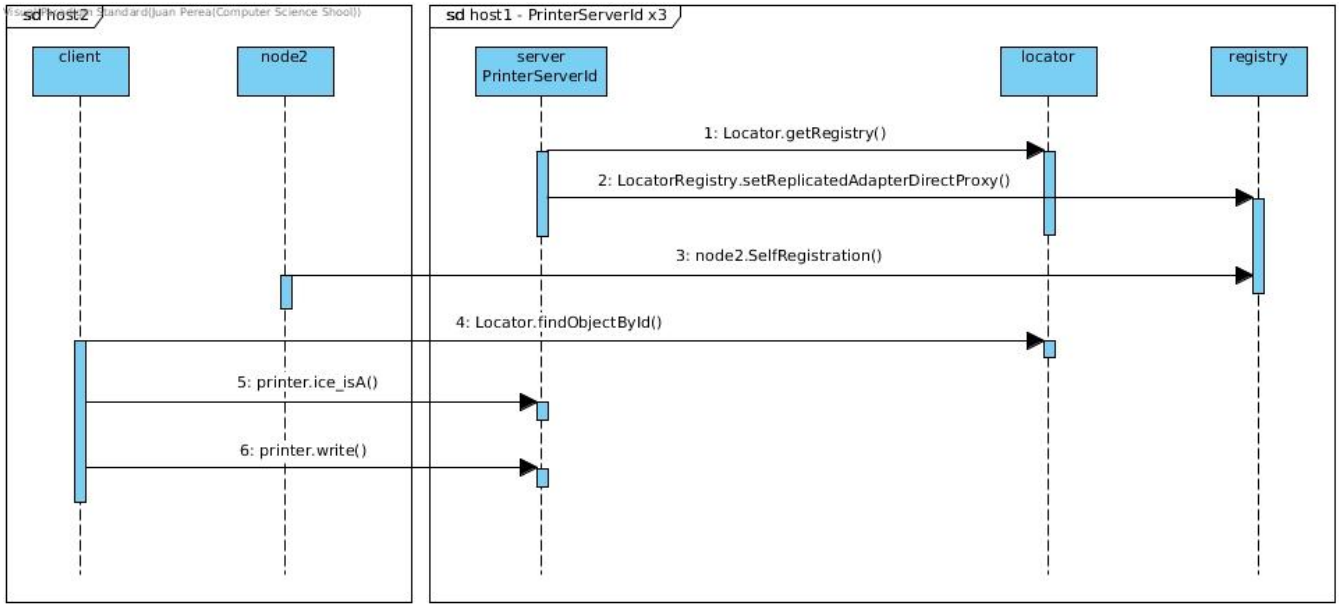
Dento del **host1**, tengo en ejecución el **node1** que a su vez contiene una instancia de *PrinterAppIcePatch2*, creada con la plantilla por defecto de IcePatch2, que será la encargada de distribuir los binarios de la aplicación **PrinterApp** a todos aquellos nodos que forman parte del sistema distribuido y, en mi caso, tres instancias diferentes de **PrinterServer{1-3}** que han sido creados usando una server template personalizada llamada de la misma forma: *PrinterServer*. Dicha plantilla genera a su vez el adaptador de objetos para cada server incluyendo el nombre del objeto bien conocido *printer{1-3}* para su registro en el *registry*, su tipo que es *Printer* y la propiedad *Identity* usada para poder hacer referencia a la identidad del objeto desde el servidor. Todos se activan bajo demanda (5s).

Todas las instancias de *PrinterServer* pertenecen a un *replica group* llamado **ReplicatedPrinterAdapter** teniendo dicho adaptador de objetos un objeto bien conocido *printer* del tipo *Printer* con una política de balanceo de carga aleatoria; podremos pues ejecutar clientes llamando al grupo de réplicas y la elección del servidor se realizará de forma transparente al cliente.

La interfaz gráfica de usuario de *IceGrid* está también ejecutada en el primero de los nodos para poder administrar toda la aplicación y realizar el despliegue pertinente.

En el segundo **host2** previamente mencionado, tengo en ejecución un **node2** cuya configuración apunta a la ip del host1 para que se pueda establecer correctamente la conexión entre ambos hosts y aparezca el nodo en la interfaz, en ella podemos apreciar en las propiedades que efectivamente está corriendo en un sistema debian 4.9.1. Las ejecuciones del cliente se realizan a su vez desde éste segundo host, apuntando también la configuración del locator a la ip del host1 para que se puede comunicar con el resgitry y poder así preguntar los correspondientes proxies directos de los endpoints de forma transparente para nosotros que solamente mencionaremos al grupo de réplicas creado.

He decidido a su vez realizar un diagrama de secuencia al igual que en la primera entrega con la finalidad de ver de forma muy simplificada como sería la comunicación en este caso. Únicamente mencionar que el *node2.SelfRegistration()* incluiría muchos más mensajes, no es una tarea tan simple como la he representado y que ahora se utilizan las llamadas **setReplicatedAdapterDirectProxy(id, replicagroup, proxy)** para registrar los endpoints en el registry de forma que pertenezcan a un grupo de réplicas y **findObjectById(Identity id)** para encontrar un objeto por identidad (printer en nuestro caso). Anteriormente las llamadas equivalentes de las que hacíamos uso eran **setAdapterDirectProxy(id, proxy)** y **findAdapterbyId(id)** respectivamente.



1.2 VIDEO MOSTRANDO EL SISTEMA EN FUNCIONAMIENTO

- Está subido a Dropbox, puede ser visto aquí.

*enlace: https://www.dropbox.com/s/p045hdujwlkx0i0/video.mp4